Chapter 6: CPP Arrays, Vectors and Stacks

Contents

Time required: 90 minutes

DRY

Don't Repeat Yourself

Online Tutorial

Go through the following tutorials before starting the tutorials

- C++ Arrays
- C++ Arrays Video
- Vectors (Tutorial with Try it Live)
- Vectors and Vector Functions (Video)

Tutorial 1: Stack Frames

Stack frames are an area in memory used to store function calls and their local variables. The stack is comprised of several Stack Frames, with each frame representing a function call.

The size of the stack increases in proportion to the number of functions called, and then shrinks upon return of a completed function.

The Stack works on a LIFO (Last In First Out) basis.

Each stack frame contains:

1. The returning line number
2. Any arguments from the called function
3. Storage space for all the function's (automatic) variables
4. (various bookkeeping data)

Consider the following program, containing two functions:

```
#include <stdio.h>

// Function prototypes
void firstFunc();
void secondFunc();

int main()
{
    printf("Hello, World! \n");
    printf("Main is the first function in the stack.\n");
    firstFunc();
        printf("The stack will be empty when the program exits.");
    return 0;
}

void firstFunc()
{
    int myInt = 17;
    printf("This is the second function in the stack.\n");
    printf("This function has an int with a value of: %d \n", myInt);
    secondFunc();
    printf("See you later.");
}

void secondFunc()
{
    int yourInt = 42;
    char myChar = 'd';
    printf("This is the third function in the stack.\n");
    printf("This function has an int: %d, and a char: %c \n", yourInt,
myChar);
}
```

Stack frames

1. Upon program start an initial stack frame is created for main()
2. firstFunc() is called and a new stack frame is created from unused stack memory, containing:

Line to return to = the line after where it was called from in main() = Line 9

Storage space for an int

3. secondFunc() is called and a new stack frame is created from unused stack memory, containing:

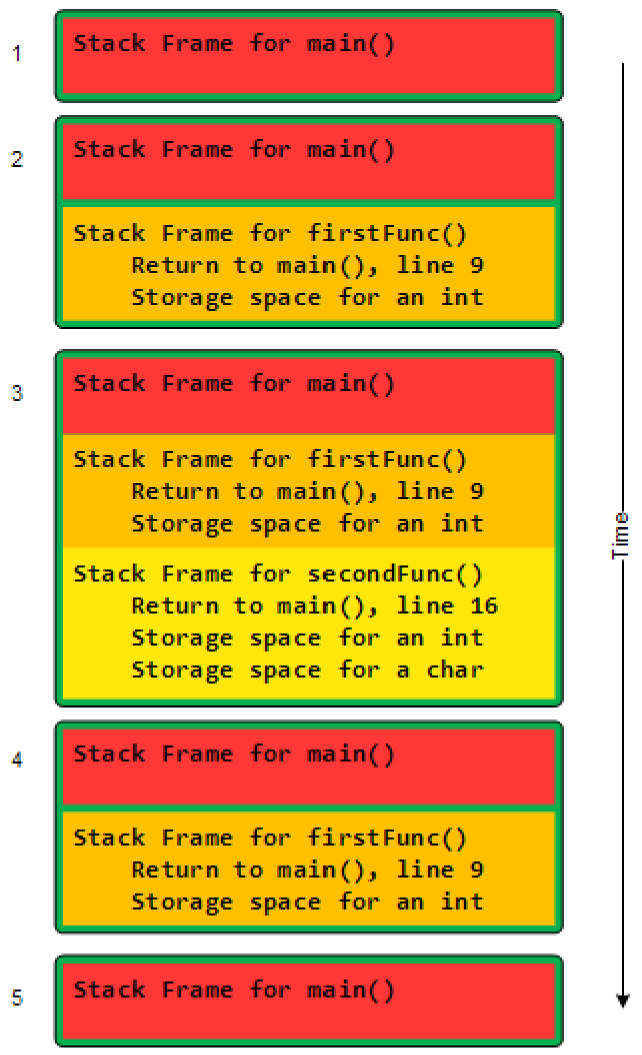
Line to return to = the line after where it was called from in firstFunc() = Line 15

Storage space for an int

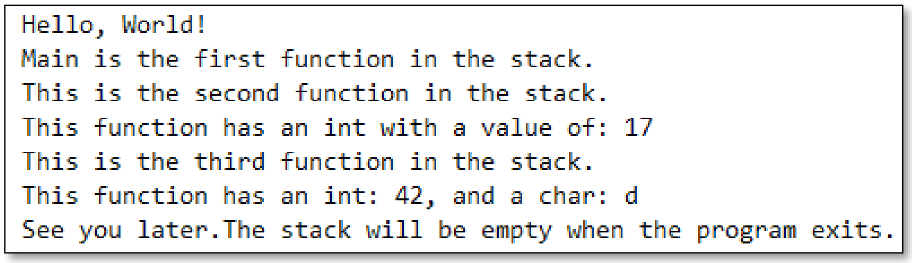
Storage space for a char

4. When secondFunc() returns, it's frame is used to determine where to return to (line 15 of firstFunc()), then deallocated and the space returned to the stack
5. When firstFunc() returns, it's frame is used to determine where to return to (line 9 of main()), then deallocated and the space returned to the stack

When main() returns, the program ends.



Example run:

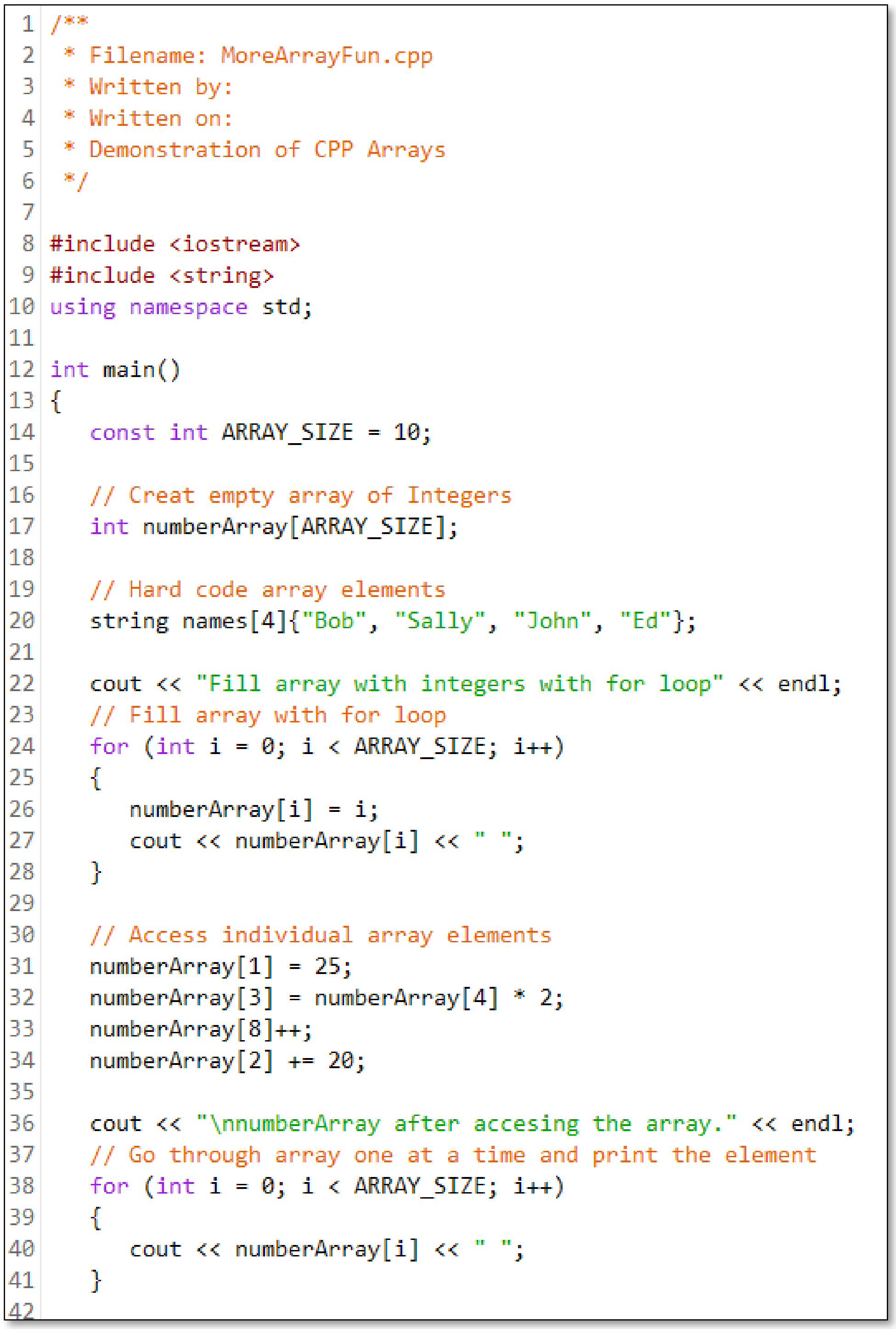


```
Hello, World!
Main is the first function in the stack.
This is the second function in the stack.
This function has an int with a value of: 17
This is the third function in the stack.
This function has an int: 42, and a char: d
See you later.The stack will be empty when the program exits.
```

Please run the above program. Attach a screen shot showing the run.

Tutorial 2: MoreArrayFun

Arrays hold multiple values. This example shows a couple of ways to initialize and iterate through an array.



```cpp
/**
 * Filename: MoreArrayFun.cpp
 * Written by:
 * Written on:
 * Demonstration of CPP Arrays
 */

#include <iostream>
#include <string>
using namespace std;

int main()
{
   const int ARRAY_SIZE = 10;

   // Creat empty array of Integers
   int numberArray[ARRAY_SIZE];

   // Hard code array elements
   string names[4]{"Bob", "Sally", "John", "Ed"};

   cout << "Fill array with integers with for loop" << endl;
   // Fill array with for loop
   for (int i = 0; i < ARRAY_SIZE; i++)
   {
      numberArray[i] = i;
      cout << numberArray[i] << " ";
   }

   // Access individual array elements
   numberArray[1] = 25;
   numberArray[3] = numberArray[4] * 2;
   numberArray[8]++;
   numberArray[2] += 20;

   cout << "\nnumberArray after accesing the array." << endl;
   // Go through array one at a time and print the element
   for (int i = 0; i < ARRAY_SIZE; i++)
   {
      cout << numberArray[i] << " ";
   }

```

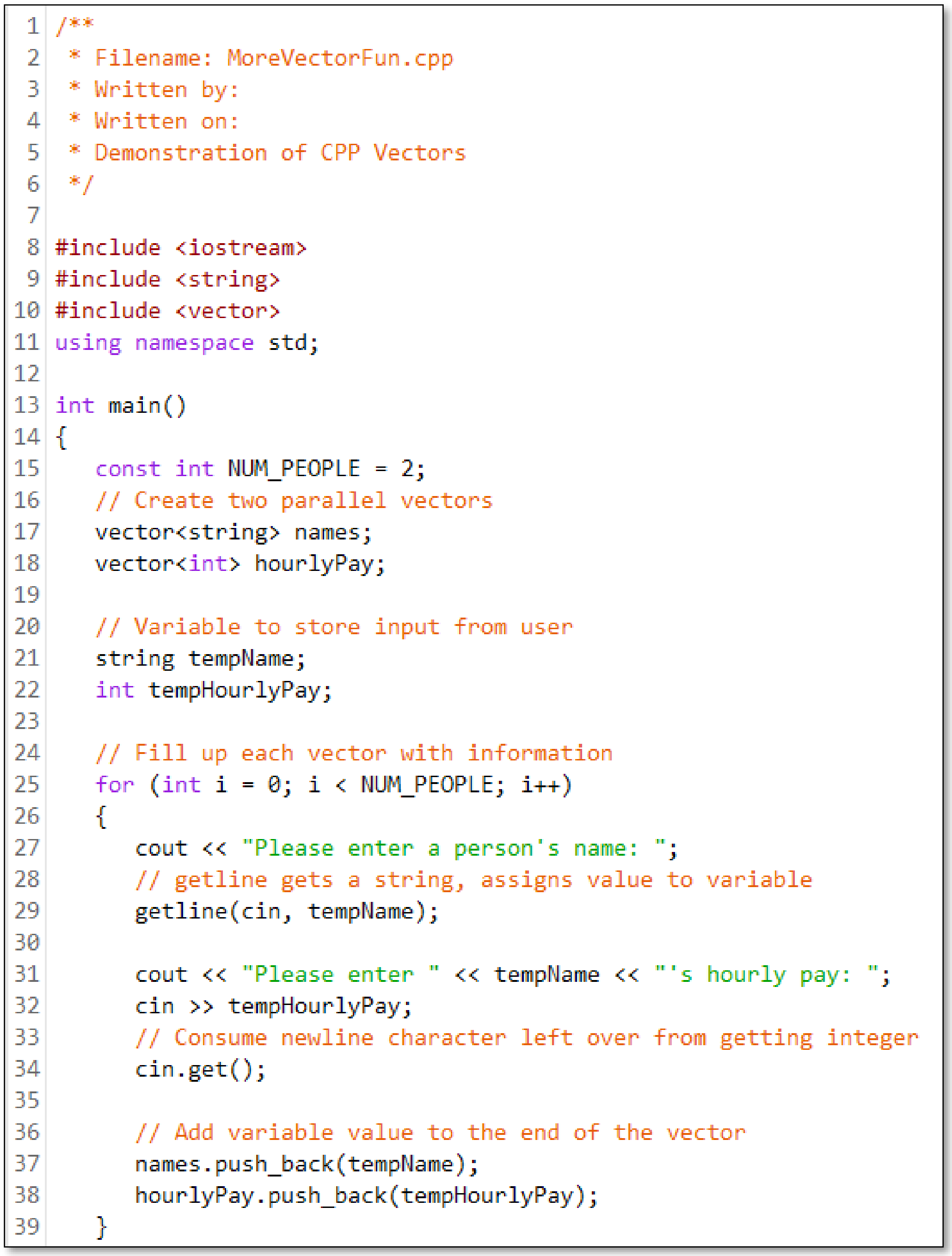
```cpp
    cout << "\nFor each loop with string array" << endl;
    // For Each loop goes through each element in an array
    for (string a : names)
    {
        cout << a << " ";
    }
    return 0;
}
```

Example run:

```
Fill array with integers with for loop
0 1 2 3 4 5 6 7 8 9
numberArray after accesing the array.
0 25 22 8 4 5 6 7 9 9
For each loop with string array
Bob Sally John Ed
```

Tutorial 3: MoreVectorFun

Vectors are much like arrays. Vectors are mutable, which means that they can be changed after they are created.



```cpp
/**
 * Filename: MoreVectorFun.cpp
 * Written by:
 * Written on:
 * Demonstration of CPP Vectors
 */

#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main()
{
   const int NUM_PEOPLE = 2;
   // Create two parallel vectors
   vector<string> names;
   vector<int> hourlyPay;

   // Variable to store input from user
   string tempName;
   int tempHourlyPay;

   // Fill up each vector with information
   for (int i = 0; i < NUM_PEOPLE; i++)
   {
      cout << "Please enter a person's name: ";
      // getline gets a string, assigns value to variable
      getline(cin, tempName);

      cout << "Please enter " << tempName << "'s hourly pay: ";
      cin >> tempHourlyPay;
      // Consume newline character left over from getting integer
      cin.get();

      // Add variable value to the end of the vector
      names.push_back(tempName);
      hourlyPay.push_back(tempHourlyPay);
   }
```

```cpp
    // Create separation between input and output
    cout << endl
        << endl;

    // Iterate through vector to display information
    for (int i = 0; i < NUM_PEOPLE; i++)
    {
        cout << names[i] << "'s hourly pay is " << hourlyPay[i] << endl;
    }

    return 0;
}
```

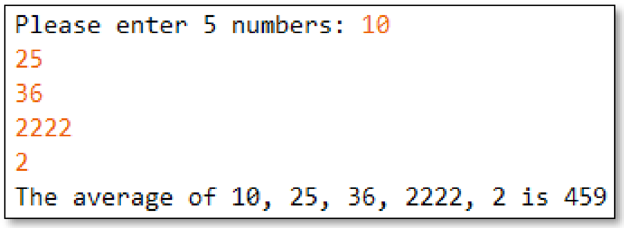
Example run:

```
Please enter a person's name: Joan
Please enter Joan's hourly pay: 12
Please enter a person's name: Laurie
Please enter Laurie's hourly pay: 12


Joan's hourly pay is 12
Laurie's hourly pay is 12
```

Tutorial 4: Vector Average

```cpp
/**
 * Filename: VectorAverage.cpp
 * Written by:
 * Written on:
 * Average a vector of numbers
 */

#include <iostream>
#include <vector>
using namespace std;

int main()
{
    double sum = 0.0;
    double average = 0.0;
    const int NUMBER_OF_ENTRIES = 5;
    vector<double> numbers(NUMBER_OF_ENTRIES);

    cout << "Please enter " << NUMBER_OF_ENTRIES << " numbers: ";

    // Allow the user to enter in the values
    for (int i = 0; i < NUMBER_OF_ENTRIES; i++)
    {
        // Insert the number into the vector
        cin >> numbers[i];
        // Keep a running total
        sum += numbers[i];
    }

    // Calculate the average
    average = sum / NUMBER_OF_ENTRIES;

    // Display the results
    cout << "The average of ";
    for (int i = 0; i < NUMBER_OF_ENTRIES - 1; i++)
        cout << numbers[i] << ", ";

    cout << numbers[NUMBER_OF_ENTRIES - 1] << " is "
              << average << "\n";

    return 0;
}
```

Example run:



```
Please enter 5 numbers: 10
25
36
2222
2
The average of 10, 25, 36, 2222, 2 is 459
```

Assignment Submission

Submit all C++ code files and a screenshot of each program showing that they work.